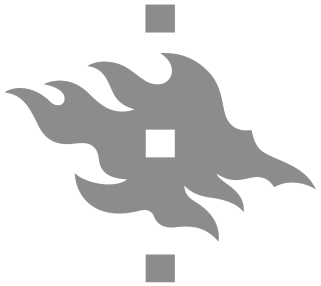
Calculating finite-differences with GPUs

Discussing an implementation called

Astaroth

by Miikka Väisälä (University of Helsinki)
and Johannes Pekkilä (Aalto University)

13.5.2015 @ NTNU



UNIVERSITY OF HELSINKI

Introduction

- Graphics Processing Units (GPUs) as tools for scientific high-performance computing have been gaining increasing attention.
- GPUs promise increased computational performance, so far, with the cost of more complex programming.
- Personally, I have worked with CUDA C since 2012, and I would like to share some experiences and insights.

Background

- The only way to understand how GPUs could be used for computational MHD, is to actually program something.
- CUDA C was chosen over CUDA Fortran because CUDA C is more available, free and up to date.
- After a series of wrong choices, dead ends and rewrites, our unholy baby was born...

Astaroth

- The code utilizes *6th order finite differences* and *2N-Runge-Kutta scheme*.
- At the moment, the code supports *isothermal hydrodynamics*.
- Experimental and under development.

"He is a Mighty, Strong Duke and appeareth in the Form of an hurtful Angel riding on an Infernal Beast like a Dragon, and carrying in his right hand a Viper. He giveth true answers of things Past, Present, and to Come, and can discover all Secrets. He ruleth 40 Legions of Spirits."
– Ars Goetia, S. L. MacGregor Mathers' translation (1904)

CUDA basics

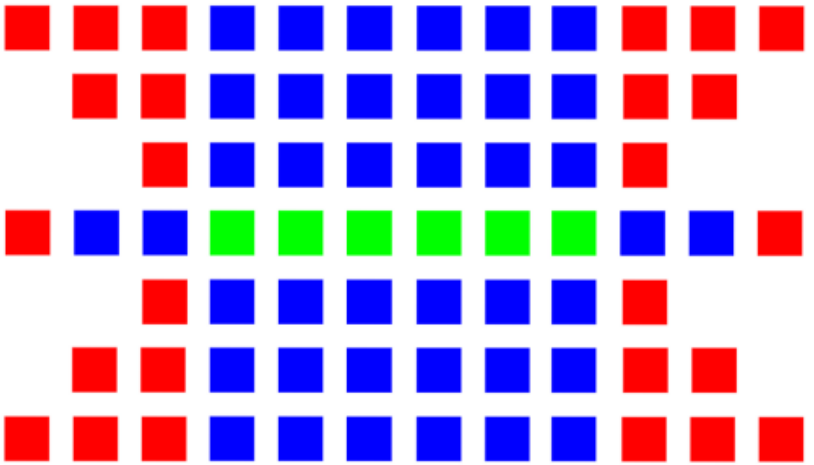
- *CUDA* stands for *Compute Unified Device Architecture*
- CUDA *devices* (GPUs) contain several parallel streaming multiprocessors.
- *Compute capability* version determines the basic capabilities of the device.
- Parts of the computation jobs are divided into *blocks*.
- A block contains a number of *threads*.
- Threads within a block are computed simultaneously by default.
- Memory management is a complicated business.

CUDA kernels

- GPU operations are handles by *kernel* functions.
- `__global__` kernels can be invoked by the *host*.
  Eg: `sample_kernel<<<blocksPerGrid, threadsPerBlock>>>( d_array, … );`
- `__device__` kernels only operate within a thread, and act otherwise like any C style function.

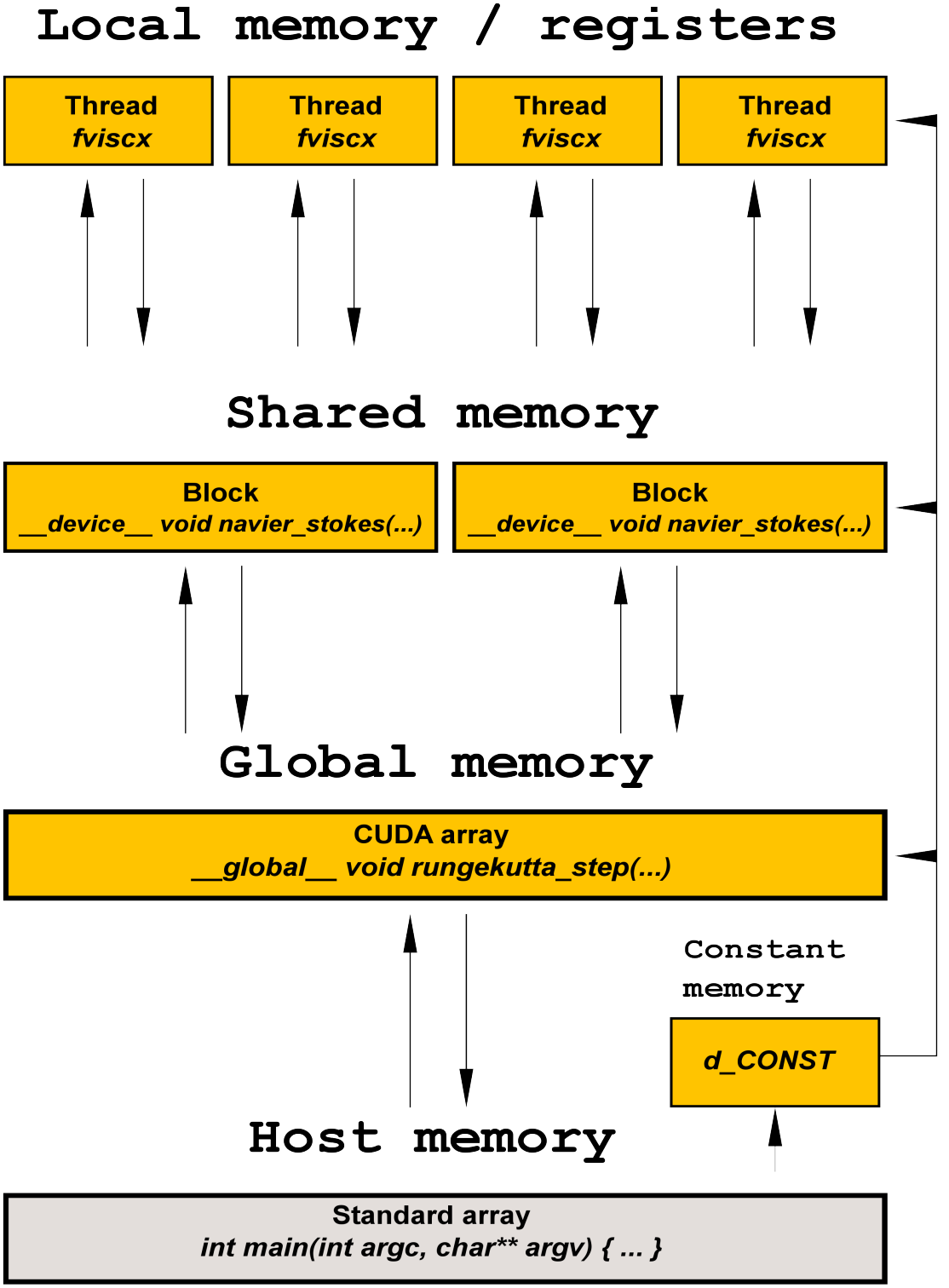
Our basic methods

- Variable arrays are operated within the *global memory* of the GPU device.
- Derivatives are always computed in the *shared memory*, because it has less latency than the global memory.
- A shared memory block contains all points needed by the derivatives.
- Variables which stay constant during a time step are computed host-side and used from the *constant memory* in device operations.



Memory in Astraroth

(and CUDA in general)

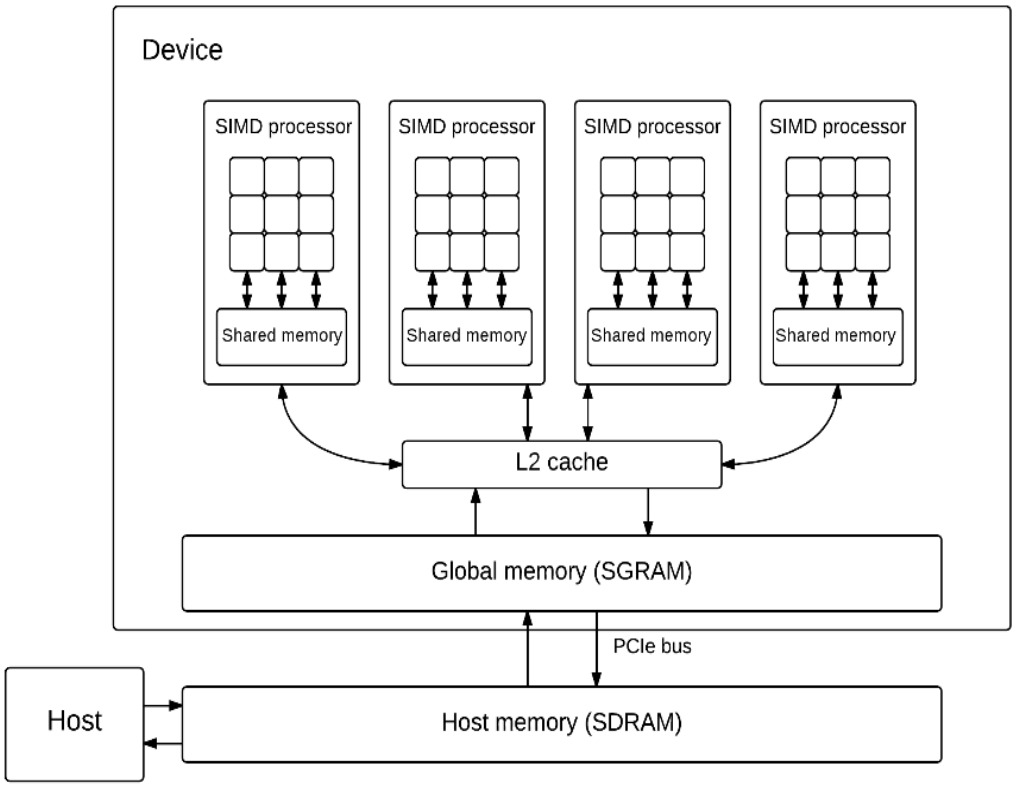


Challenges: conceptual

- You need to *think with threads not loops*: if you include a loop within a CUDA kernel, be very careful.
- You need to *pay attention to the hardware* to optimize reasonably.
- *Memory management takes a lot of work*: coding can become easily pretty complicated and difficult to read.

Challenges: hardware

- *CUDA supports only Nvidia hardware*: limits the portability.
- *Shared memory space is very limited*. 48 KB per thread block is very hard limit when you need many points for finite differences.
- It is perhaps *impossible to avoid some memory issues*.
- *Very large grids will require multiple GPU:s*. 256^3 resolution is still ok for modern Tesla devices.



Challenges: testing and debugging

- *Complex memory management can easily create less than obvious programming errors*, and you might not get a warning.
- *Multiple simultaneous threads make finding errors challenging*.
- *Debugging tools are very limited*. E.g. cuda-gdb is very cumbersome.
- *Synchronization needs special attention*. Danger to overwrite data that is still in use.

Why CUDA/GPUs?

- Nvidia devices are *popular in CPGPU computing*.
- *The speed-up can be significant*. Might be worth the effort in long term.
- OpenACC code is still difficult to optimize in satisfying way for complicated problems. With CUDA *you know what it does*.
- But *might be too much work*. You might want to wait.

Future

- Using Astaroth for actual science.
- Other ways of memory handling in consideration.
- Using what learned in the GPU-implementation of the Pencil Code. (?)

Thank you for listening!